HIGH SPEED AND COMPACT OVERFLOW DETECTION FOR SHIFTERS

Rimon Ikeno

TECHNICAL FIELD OF THE INVENTION

The technical field of this invention is overflow detection for shifter circuits used in data processing.

BACKGROUND OF THE INVENTION

Detection of overflow when shifting data in an arithmetic logic unit (ALU) requires logic with significant propagation delay. Often overflow detection takes more time than the shift operation that generates the overflow. The conventional algorithm for overflow detection performs mask generation and data propagation serially or sequentially. This algorithm may

be described as follows. First a shift mask is generated from the binary value of the desired impending shift.

Table 1 shows an example of the shift mask generation for 16-bit shifter. The mask has five leading bits '1', ten following bits '0' and a least significant bit that is a don't care (X).

Shift value	0101
Mask	1111 1000 0000 000X

Table 1

In the next step the mask is used to filter the data with AND gates as follows. The least significant bit of the data is ignored because it will never be shifted out by shift operation. Table 2 shows 16-bit filtering for the example shift value of 5 (binary 0101).

Data	0010 1001 1110 0011
Shifted Data	0011 1100 011X XXXX
Mask	1111 1000 0000 000X
Result	0010 1000 0000 000X

Table 2

The resulting bit sequence contains '1' bits if the shift would cause an overflow. The propagation circuit detects occurrence of '1' bits by taking a logical OR of each bit in the sequence, and '1' appears at the overflow output OVF when overflow occurs.

This conventional algorithm takes significant time to execute because data propagation for the OR operation starts after the mask generation and masking operation complete. The truth table for 16-Bit overflow detection is given in Table 3. The complexity of a conventional 16-bit overflow detector function is not extraordinary and the truth table may be satisfied with a straightforward logic design.

Shift Value	LSB Data='1' Causing Overflow with Shift	Masked Bits	S0	S1	S2	S3
15	D1	D15–D1	1	1	1	1
14	D2	D15–D2	0	1	1	1
13	D3	D15–D3	1	0	1	1
12	D4	D15–D4	0	0	1	1
11	D5	D15–D5	1	1	0	1
10	D6	D15–D6	0	1	0	1
9	D7	D15–D7	1	0	0	1
8	D8	D15–D8	0	0	0	1
7	D9	D15–D9	1	1	1	0
6	D10	D15-D10	0	1	1	0
5	D11	D15-D11	1	0	1	0
4	D12	D15-D12	0	0	1	0
3	D13	D15-D13	1	1	0	0
2	D14	D15-D14	0	1	0	0
1	D15	D15	1	0	0	0

Table 3

Figure 1 illustrates a conventional overflow detection circuit for a 16-bit shifter. The circuit consists of three parts: a mask decoder generator at levels 101, 102, and 103; masking levels 104 and 105; and a propagation stage in levels 106 through 108. Shift value 100 is the number of bit positions the data is to be shifted during the impending shift in binary. The mask decoder generator at levels 101, 102 and 103 decodes the value into a series of binary digits called the shift mask S. When the shift value S is N, the shift mask consists of N bits of '1's and M-N-1 bits of '0's, where M is bit-length of the data. In the example illustrated above, with shift value binary '0101' (or decimal 5), the leading five bits D11 through D15 mark bit positions in which a '1' in the data produces an overflow. Mask generation is performed in logic levels 101 through 103. The 15-bit shift mask appears at the output of logic level 103. Recall that the least significant bit cannot generate an overflow. Then, a cluster of AND gates performs the masking operation driving outputs at level 104. The logic masks these bit positions in logic levels 101 through 103. Data information enters at level 104 and the resulting bit sequence from the masking operation enters at level 105. The remaining logic levels 106, 107, and 108 form an OR-tree to compute the presence of a data value of '1' within the masked field producing an overflow.

Figures 2A and 2B illustrate a conventional 32-bit overflow detector. Figure 2A is the first portion and Figure 2B the second portion of the logic. First note that several packets of signals form the interconnect between the two figures. Signal packet 201 passes the five shift bits S0 through S4 between the two drawings. Signal packet 202 passes several intermediate

signals generated in Figure 2B to inputs of logic in Figure 2A. Signal packet 203 passes the sixteen most significant data bits D31:D16 from Figure 2A to Figure 2B. Finally, two inputs 206 and 207 to OVF output gate 208 of Figure 2A come from logic generating these signals in Figure 2B.

Table 4 shows the truth table for the 32-bit overflow detector function for shifters. This table can be applied directly to generation of the logic of Figures 2A and 2B which are most similar in organization to that of the conventional 16-Bit shifter overflow detector function of Figure 1. It is worthwhile to point out that in the design of many high speed logic functions optimal propagation delay performance dictates that each gate have a relatively small number of inputs. Often it is desirable to use cascaded two input gats in preference to less levels of gates have a large number of inputs (e.g. 8-input gates). Also it is sometimes preferable to use cascaded NAND gates to implement the logical equivalent of and AND-OR function for example. The cascaded NAND function appears in several parts of the logic of Figures 2A and 2B. One example is noted with NAND gates 211, 212, and 213 cascaded with NAND gate 205. Notice that in both the conventional 16-bit overflow function of Figure 1 and the conventional 32-bit shifter of Figures 2A and 2B, decoding of the shift value precedes the input of data in the logic path. Levels 101, 102 perform the shift decoding in Figure 1. Levels 201 and 202 perform shift value decoding in Figures 2A and 2B.

Shift Value	LSB Data ='1'Causing Overflow with Shift	Masked Bits	S0	S1	S2	S3	S4
31	D1	D31-D1	1	1	1	1	1
30	D2	D31-D2	0	1	1	1	1
29	D3	D31-D3	1	0	1	1	1
28	D4	D31-D4	0	0	1	1	1
27	D5	D31-D5	1	1	0	1	1
26	D6	D31-D6	0	1	0	1	1
25	D7	D31-D7	1	0	0	1	1
24	D8	D31-D8	0	0	0	1	1
23	D9	D31-D9	1	1	1	0	1
22	D10	D31-D10	0	1	1	0	1
21	D11	D31-D11	1	0	1	0	1
20	D12	D31-D12	0	0	1	0	1
19	D13	D31-D13	1	1	0	0	1
18	D14	D31-D14	0	1	0	0	1
17	D15	D31-D15	1	0	0	0	1
16	D16	D31-D16	0	0	0	0	1
15	D17	D31-D17	1	1	1	1	0
14	D18	D31-D18	0	1	1	1	0
13	D19	D31-D19	1	0	1	1	0
12	D20	D31-D20	0	0	1	1	0
11	D21	D31-D21	1	1	0	1	0
10	D22	D31-D22	0	1	0	1	0
9	D23	D31-D23	1	0	0	1	0
8	D24	D31-D24	0	0	0	1	0
7	D25	D31-D25	1	1	1	0	0
6	D26	D31-D26	0	1	1	0	0
5	D27	D31-D27	1	0	1	0	0
4	D28	D31-D28	0	0	1	0	0
3	D29	D31-D29	1	1	0	0	0
2	D30	D31-D30	0	1	0	0	0
1	D31	D31	1	0	0	0	0

Table 4

SUMMARY OF THE INVENTION

This invention describes a unique high-speed implementation for overflow detection logic to be used in high performance shifter functions. The overflow logic makes use of parallelism in combining shift value decoding and mask generation logic with the logic necessary to propagate data. Designs for both 16-bit and 32-bit shifters are presented and performance improvement of the new designs over conventional overflow detection circuits is demonstrated.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other aspects of this invention are illustrated in the drawings, in which:

Figure 1 illustrates a conventional overflow detection circuit for a 16-bit shifter (Prior Art);

Figure 2A illustrates the first portion of a conventional overflow detection circuit for a 32-bit shifter (Prior Art);

Figure 2B illustrates the second portion of a conventional overflow detection circuit for a 32-bit shifter (Prior Art);

Figure 3 illustrates the overflow detection circuit for a 16-bit shifter according to this invention;

Figure 4A illustrates the first portion (bits 0 through 15 and output portion) of the 32-bit shifter overflow detection circuit according to this invention;

Figure 4B illustrates the second portion (bits 16 through 31) of the 32-bit shifter overflow detection circuit according to this invention;

Figure 5 illustrates the implementation details of the two-to-one multiplexers for the 32-bit shifter overflow detection circuit according to this invention; and

Figure 6 illustrates the implementation details of the eight-to-one multiplexers for the 32-bit shifter overflow detection circuit according to this invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

In the shift overflow detection circuits of this invention the masking operation is performed in parallel with logical OR operations on the data. This reduces the number of delay levels in the critical path and the total propagation delay time of the circuit. The number of circuit elements is reduced as well, resulting in less power consumption of the circuit.

Figure 3 illustrates a detection circuit for 16-bit shifter constructed by this new circuit concept. The circuit consists entirely of OR gates (312, 314, 316, 318, 329, 322, 324, 334, 338, 342, 358, 381, 384 and 393) and multiplexers (311, 313, 315, 317, 319, 321, 323, 325, 331, 332, 335, 336, 339, 340, 343, 344, 351, 352, 354, 359, 361, 365, 371, 372, 374, 378). All multiplexers have one control line each driven from one of the shift value lines 302. Multiplexers aligned prior to level 303 have select input S0. Multiplexers aligned prior to level 304 have select input S1. Level 305 select input is S2; level 306 select input is S3. Data propagation and data masking are executed simultaneously. In parallel with the data propagation by the OR gates, the multiplexers perform the masking operation following the shift values as illustrated in Figure 3. The 16-bit data 301 is combined according to the truth table of Table 1 at levels 303, 304, 305 and 306. These 4-stage paths of 2-input OR gates and 2:1 multiplexers form a tree in which the 16-bit data is compressed into a 4-bit word D 310 at level 306. The four bits of D are combined in two 2-bit OR gates 381 and 384

with two outputs at level 307. A final OR gate 393 at level 308 reduces these two to the output overflow signal OVF 309 which is high if an overflow has been detected.

The OR gates and multiplexers are arranged and connected following the four construction rules below. The following description uses these definitions: M is the bit-length of the data; N is the bit length of shift value; ELEM(i,d) is the circuit element placed at d-th bit of i-th stage; OUT(i,d) is the output of ELEM(i,d); OR(A,B) represents an OR gate with signals A and B as inputs; MUX(A,B,i) represents a multiplexer that propagates signal A when the i-th bit of the shift value is 1 and propagates B when i-th bit of the shift value is 0.

Rule 1: If $d=2^n*2^i$ and n is in the range $1 \leq n \leq ((M/2^{i+1})-1)$, then ELEM(i,d) = OR(OUT(i-1,d+2^i), OUT(i-1,d)).

Rule 2: If $d=2^n*2^i+2^j$, n is in the range $1 \leq n \leq ((M/2^{i+1})-1)$ and j is in the range $0 \leq j \leq i-1$, then ELEM(i,d) = MUX(OUT(i-1,d), OUT(i-1,d+2^i), i).

Rule 3: If $d=(2^n+1)*2^i$, n is in the range $1 \leq n \leq ((M/2^{i+1})-1)$ and j is in the range $0 \leq j \leq i-1$, then ELEM(i,d) = MUX(OUT(i-1,d), 0, i).

Rule 4: For all other combinations of i and d, no element at location ELEM(i,d).

The number of stages in this circuit is smaller than the number in a conventional circuit. Assume that N is the maximum bit-length of shift value and that M (= 2^N) is the bit-length of

data to be shifted. Assume the circuit includes only one and two input logic gates. The conventional circuit requires at least $\log_2(N)$ stages to construct mask generation circuit. The circuit also requires one stage of NAND gates for data masking. The propagation circuit requires $\log_2(M)=N$ stages of NOR tree. Consequently, the conventional circuit requires $N+\log_2(N)+1$ gates to detect overflow. On the other hand, in the detection circuit of this invention the multiplexer tree needs only N stages. The logical OR of the multiplexed signals requires an additional $\log_2(N)$ stages. Thus, this circuit of this invention requires $N+\log_2(N)$ gates. Thus the circuit of this invention can always be constructed with at least one less gate stage than the conventional circuit. The conventional 16-bit circuit illustrated in Figure 1 requires 63 circuit elements, while the inventive circuit illustrated in Figure 3 requires only 40. This reduction number of circuit elements will result in reduction of area of the circuit block and also in reduction of operational power.

To illustrate the extension of the algorithm given in the four rules above, a 32-bit overflow detection circuit was designed and then simplified and reduced keeping the same logic functionality. Figures 4A and 4B illustrate the resulting circuit which utilizes 2-input multiplexers (see 421) illustrated in Figure 5. The 32-bit overflow detection circuit also makes use of a more complex 8-input multiplexer 417 illustrated in Figure 6.

Referring to Figure 4A, data inputs 415 to the overflow detection circuit are combined with shift inputs 400 and 410

according to the truth table (Table 4) requirements to generate the overflow signal OVF 409. The 32-bit implementation calls for a large number of wide-OR logic gates that are more efficiently implemented by NAND gates (482, 484, 485, 487, 488, 490, 491) combined with multiplexer stages to form the equivalent OR terms. For purposes of propagation delay analysis, the logic levels are labeled as 401 through 404, three delays in total prior to the output stage OR gate 420. The select signals s0 and s1 are passed to the higher order bits of the circuit Figure 4B as shown by signals 400 at the lower portion of Figure 4A. From the three select bits s2, s3, and s4 (signals 410 in Figure 4A), only s4 is passed through to the higher order bits of the circuit in Figure 4B (s4 is represented by 411 in Figure 4B).

Referring to Figure 4B, the higher order bits have data inputs 425, select inputs 400 and 411 and logic levels 401 through 404 similar to the labeling in Figure 4A. The higher order bits pass two signal sets back to the lower order portion of the circuit in Figure 4A. The first set is signal bundle 406 and the second is the signal 412 which forms one of the main contributors to the logic inputs for gate 420 and the output signal OVF 409 of Figure 4A.

Referring again to Figures 4A and 4B, intermediate signals 600 through 607 are generated as inputs to the 8-input multiplexer 417. The use of the 8-input multiplexer 417 at this point in the logic flow is crucial to the high performance of the 32-bit overflow detection circuit. By the nature of detection of overflow in systems having 32 bits and greater, some stages (wide-OR terms in particular) are needed that benefit greatly from the speed improvements achieved by

parallelism. This is especially true here as is illustrated in the multiplexer oriented logic as described in Figures 5 and 6.

Two multiplexer stages are illustrated in Figures 5 and 6. Figure 5 illustrates the 2-input multiplexer 500 and 501. The multiplexer-NAND implementation of the 2-input multiplexer is illustrated in 500 of Figure 5. The implementation is taken one step closer to the component level by the circuit 501 of Figure 5.

Figure 6 illustrates the 8-input multiplexer 600 broken down into two portions of the implementation. The first portion shows the shift decode portion which decodes shift inputs 610 into decoded shift inputs 612. The second portion is the 8:1 multiplexer. This combines the data inputs 600 through 607 and the decoded shift inputs 612 to form the intermediate output term 608.

Table 5 compares the performance of the circuit in Figures 4A and 4B with the conventional 32-bit overflow detector circuit illustrated in Figures 2A and 2B. Circuit performance of these circuits was simulated using an industry standard static timing analysis tool. Table 5 also compares the power consumption of the two circuits using an industry standard power dissipation analysis tool. The two circuits were designed using the same circuit library and identical models at corresponding logic levels. Analysis condition was 125 °C and a power supply of 1.35 Volts. The comparison results summarized in the table show that the new circuit provides significantly better results compared to the conventional circuit on both propagation delay performance and power dissipation.

	Conventional	Present Invention	Difference
Maximum Delay Path in Gates	6	4	-33%
Maximum Delay Path in nanoSec	0.750	0.629	16%
Power [mW/MHz]	0.501	0.336	-33%

Table 5